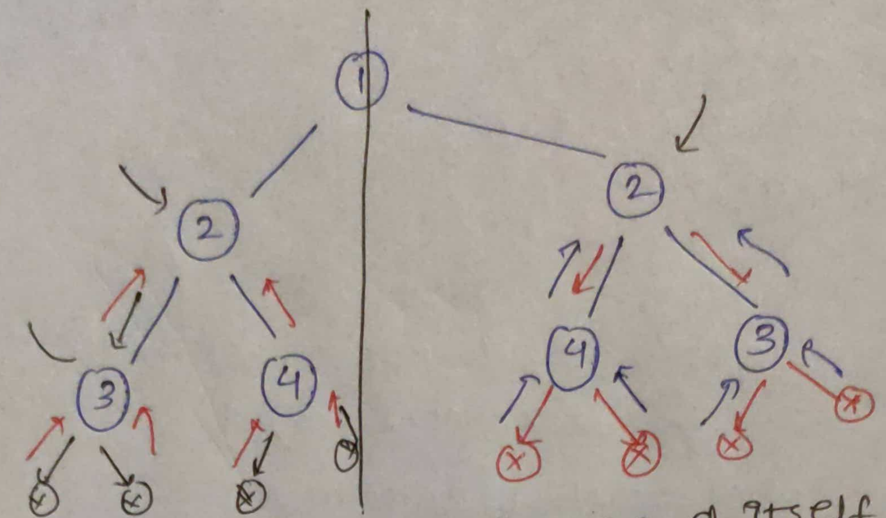
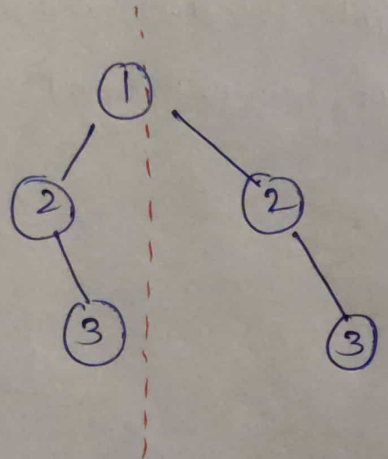
Check for Symmetric Binary Tree:-



it forms a mirror image of itself around the centre.



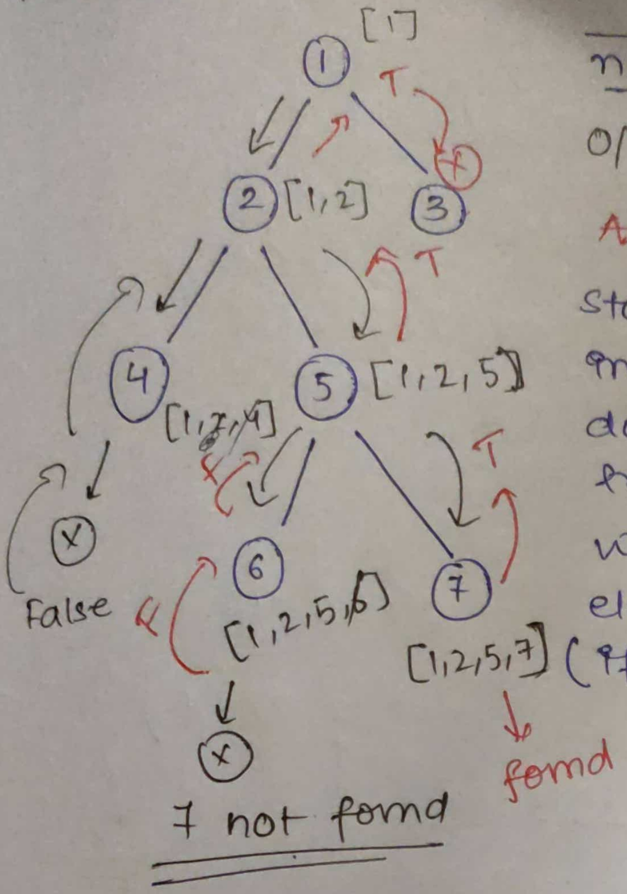
Not a mirror

Approach:- Just divide the left-subtree and right subtree and perform the pre-order traversal on both (by checking the node->val at each point).

Code:-

```
bool isSymmetric (TreeNode *root){
    return root==NULL || isSymmetricHelp(root->
                           left, root->right)
    }
bool isSymmetricHelp(Tree Node * left, * right){
        if (left=NULL || right== NULL) return l==r;
  if (left->val != right->val) return false;
  return isSymmetric Help (left -> left, right -> right) &&
            " ( left->right, right->left);
```

Root to Node Path.



node = 7.

O/P: [1 2 5 7]

Approach: Inorder traversal.

Store value of every node in array if found the desired node return true else return false, while returning pop the elements from the array (if node! not found)

Code:-

```
bool getPath (TreeNode *root, vec<int>& arr, int x){
  if(!root)
     return false;
  arr.push_back(root->val);
  if(root->val==x)
     return true;
   if (getPath (root->left, arr, x) || getPath (root->right
                                                  ,arr, x))
        return true;
   arr.pop_back();
   return false;
}
vector<int> solve (TreeNode *A, int B){
     vec<int> arr;
   if (A==NULL) return arr;
   getPath(A, arr, B);
   return arr;
}
```

$\boxed{TC: O(n)}$